Sisteme de Operare

**Nicoleta Radu – Tema 1**

# Exercitiul 1

//\*//#undef UNICODE

//\*//go to Project Properties > Advanced > Character Set and put Not Set

#include <windows.h>

#include <string>

#include <atlstr.h>

#include "resource.h"

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

//\*//DLGPROC CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE;

HWND hwndMain;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Dialog";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwnd = CreateWindow(szAppName,

"Un program simplu",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

NULL,

NULL,

hInstance,

NULL);

SetWindowPos(hwnd, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOACTIVATE);

ShowWindow(hwnd, SW\_HIDE);

UpdateWindow(hwnd);

hwndMain = hwnd;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

static HINSTANCE hInstance;

switch (iMsg)

{

case WM\_CREATE:

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) {

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, (DLGPROC)DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

unsigned int factorial(unsigned int n)

{

if (n == 0 || n == 1)

return 1;

return n \* factorial(n - 1);

}

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

char text[100];

int randomNumber{};

unsigned int factorialOfRandomNumber{};

char stringNumberBuffer[100] = "\0";

char stringFactorialBuffer[100] = "\0";;

switch (iMsg) {

case WM\_INITDIALOG:

return TRUE;

case WM\_CLOSE:

dlgActive = FALSE;

EndDialog(hDlg, 0);

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDC\_RANDOM:

// Calcularea numarului aleatoriu

randomNumber = rand() % 16;

// Includerea acestuia sub forma unui C-style string "stringNumberBuffer"

sprintf\_s(stringNumberBuffer, "%ld", randomNumber);

SetDlgItemText(hDlg, IDC\_EDIT\_NR, stringNumberBuffer);

return TRUE;

case IDC\_CALC:

// Luarea numarului aleatoriu din caseta de editare IDC\_EDIT\_NR

GetDlgItemText(hDlg, IDC\_EDIT\_NR, stringFactorialBuffer, 100);

// Verificarea daca cumva nu a fost generat un numar

if (strlen(stringFactorialBuffer) == 0)

{

MessageBox(hDlg, "Nu s-a generat niciun numar", "Eroare", MB\_OK);

}

// Calcularea factorialului folosing functia "factorial" scrisa mai sus

factorialOfRandomNumber = factorial(atoi(stringFactorialBuffer));

sprintf\_s(stringFactorialBuffer, "%ld", factorialOfRandomNumber);

SetDlgItemText(hDlg, IDC\_EDIT\_REZ, stringFactorialBuffer);

return TRUE;

case ID\_CANCEL:

dlgActive = FALSE;

MessageBox(hDlg, "Iesire!", "Ati selectat CANCEL", MB\_OK);

EndDialog(hDlg, 0);

return TRUE;

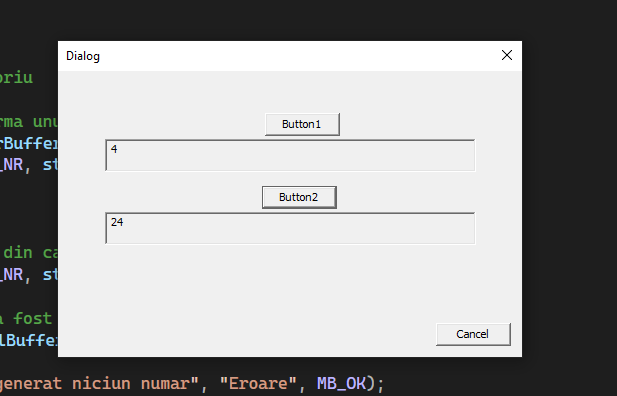
}

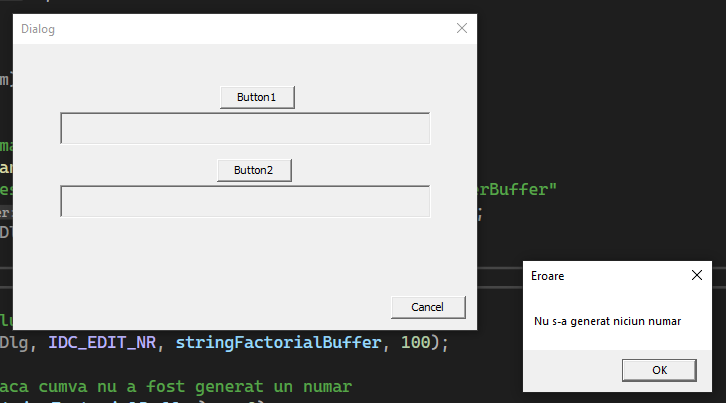
break;

}

return FALSE;

}





# Exercitiul 2

//\*//#undef UNICODE

//\*//go to Project Properties > Advanced > Character Set and put Not Set

#include <windows.h>

#include <string>

#include <atlstr.h>

#include "resource.h"

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

//\*//DLGPROC CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE; //variabila pentru casetadedialog

HWND hwndMain; //variabila handle a ferestrei principale

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Dialog";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW; //optiuni pentru stilul clasei

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

// incarca pictograma fisierelor de tip aplicatie

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW); // incarca cursorul sageata

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

// fondul ferstrei de culoare alba

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwnd = CreateWindow(szAppName, //numele clasei inregistrat cu RegisterClass

"Operatii Aritmetice", // text pentru bara de titlu a ferestrei

WS\_OVERLAPPEDWINDOW, // stilul ferestrei

CW\_USEDEFAULT, // pozitia orizontala implicitã

CW\_USEDEFAULT, // pozitia verticala implicita

CW\_USEDEFAULT, // latimea implicita

CW\_USEDEFAULT, // inaltimea implicita

NULL, // handle-ul ferestrei parinte

NULL, // handle-ul meniului ferestrei

hInstance, // proprietara ferestrei

NULL);

SetWindowPos(hwnd, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOACTIVATE);

// schimba dimensiunea, pozitia si ordinea z a ferestrei copil, a ferestrei pop-up

ShowWindow(hwnd, SW\_HIDE);

UpdateWindow(hwnd);

hwndMain = hwnd;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

static HINSTANCE hInstance;

switch (iMsg)

{

case WM\_CREATE: // operatiile ce se executa la crearea ferestrei

// se creaza caseta de dialog

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) {

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, (DLGPROC)DlgProc);

//\*//hwnd, (DLGPROC)DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

// insereaza un nou mesaj nou in coada de asteptare

// sa te inserezi tu in pizda matii si facultate de cacat

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0); // insereaza un mesaj de incheiere

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

unsigned int factorial(unsigned int n)

{

if (n == 0 || n == 1)

return 1;

return n \* factorial(n - 1);

}

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

char text[100] = "\0";

int firstNum = 0;

int secondNum = 0;

int result = 0;

char num\_1[100] = "\0";

char num\_2[100] = "\0";

char arithmetic[100] = "\0";;

switch (iMsg) {

case WM\_INITDIALOG:

return TRUE;

case WM\_CLOSE:

dlgActive = FALSE;

EndDialog(hDlg, 0);

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDC\_BUTTON\_1:

// Calcularea primului numar aleatoriu

firstNum = 1 + (rand() % 100);

sprintf\_s(num\_1, "%ld", firstNum);

SetDlgItemText(hDlg, IDC\_EDIT\_1, num\_1);

return TRUE;

case IDC\_BUTTON\_2:

// Calcularea celui de al doilea numar aleatoriu

secondNum = 1 + (rand() % 100);

sprintf\_s(num\_2, "%ld", secondNum);

SetDlgItemText(hDlg, IDC\_EDIT\_2, num\_2);

return TRUE;

case IDC\_BUTTON\_ADUNARE:

// Calcularea operatiei de adunare

GetDlgItemText(hDlg, IDC\_EDIT\_1, num\_1, 100);

GetDlgItemText(hDlg, IDC\_EDIT\_2, num\_2, 100);

// Verificarea daca cumva nu a fost generat un numar

if (strlen(num\_1) == 0 || strlen(num\_2) == 0)

{

MessageBox(hDlg, "Nu s-a generat niciun numar", "Eroare", MB\_OK);

}

// Calcularea adunarii cu numerele obtinute

result = (atoi(num\_1)) + (atoi(num\_2));

sprintf\_s(arithmetic, "%ld", result);

sprintf\_s(text, "%d + %d = %d", atoi(num\_1), atoi(num\_2), atoi(arithmetic));

MessageBox(hDlg, text, "Adunare", MB\_OK);

return TRUE;

case IDC\_BUTTON\_SCADERE:

// Calcularea operatiei de scadere

GetDlgItemText(hDlg, IDC\_EDIT\_1, num\_1, 100);

GetDlgItemText(hDlg, IDC\_EDIT\_2, num\_2, 100);

// Verificarea daca cumva nu a fost generat un numar

if (strlen(num\_1) == 0 || strlen(num\_2) == 0)

{

MessageBox(hDlg, "Nu s-a generat niciun numar", "Eroare", MB\_OK);

}

// Calcularea propriu-zisa cu numerele obtinute

result = (atoi(num\_1)) - (atoi(num\_2));

sprintf\_s(arithmetic, "%ld", result);

sprintf\_s(text, "%d - %d = %d", atoi(num\_1), atoi(num\_2), atoi(arithmetic));

MessageBox(hDlg, text, "Scadere", MB\_OK);

return TRUE;

case IDC\_BUTTON\_INMULTIRE:

// Calcularea operatiei de inmultire

GetDlgItemText(hDlg, IDC\_EDIT\_1, num\_1, 100);

GetDlgItemText(hDlg, IDC\_EDIT\_2, num\_2, 100);

// Verificarea daca cumva nu a fost generat un numar

if (strlen(num\_1) == 0 || strlen(num\_2) == 0)

{

MessageBox(hDlg, "Nu s-a generat niciun numar", "Eroare", MB\_OK);

}

// Calcularea propriu-zisa cu numerele obtinute

result = (atoi(num\_1)) \* (atoi(num\_2));

sprintf\_s(arithmetic, "%ld", result);

sprintf\_s(text, "%d \* %d = %d", atoi(num\_1), atoi(num\_2), atoi(arithmetic));

MessageBox(hDlg, text, "Inmultire", MB\_OK);

return TRUE;

case IDC\_BUTTON\_IMPARTIRE:

// Calcularea operatiei de impartire

GetDlgItemText(hDlg, IDC\_EDIT\_1, num\_1, 100);

GetDlgItemText(hDlg, IDC\_EDIT\_2, num\_2, 100);

// Verificarea daca cumva nu a fost generat un numar

if (strlen(num\_1) == 0 || strlen(num\_2) == 0)

{

MessageBox(hDlg, "Nu s-a generat niciun numar", "Eroare", MB\_OK);

}

// verificare daca se efectueaza impartirea la 0

if ((atof(num\_2) == 0))

{

MessageBox(hDlg, "Impartire la 0", "Impartire", MB\_OK);

break;

}

// Calcularea propriu-zisa cu numerele obtinute

result = (atof(num\_1)) / (atof(num\_2));

sprintf\_s(arithmetic, "%ld", result);

sprintf\_s(text, "%d / %d = %f", atoi(num\_1), atoi(num\_2), atof(arithmetic));

MessageBox(hDlg, text, "Impartire", MB\_OK);

return TRUE;

case ID\_CANCEL:

dlgActive = FALSE;

MessageBox(hDlg, "Iesire!", "Ati selectat CANCEL", MB\_OK);

EndDialog(hDlg, 0);

return TRUE;

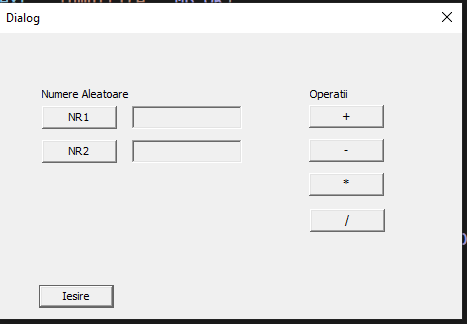
}

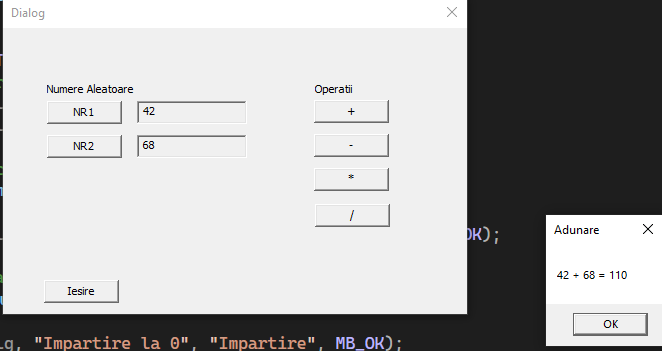
break;

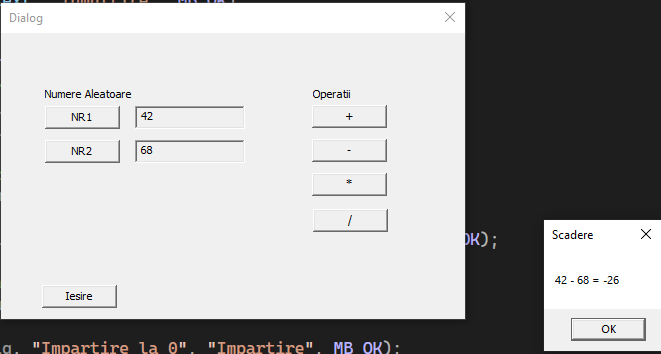
}

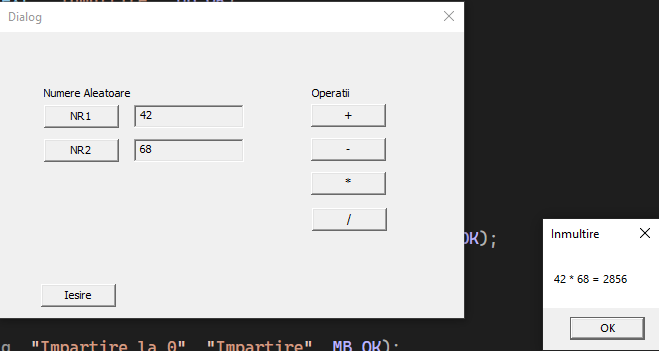
return FALSE;

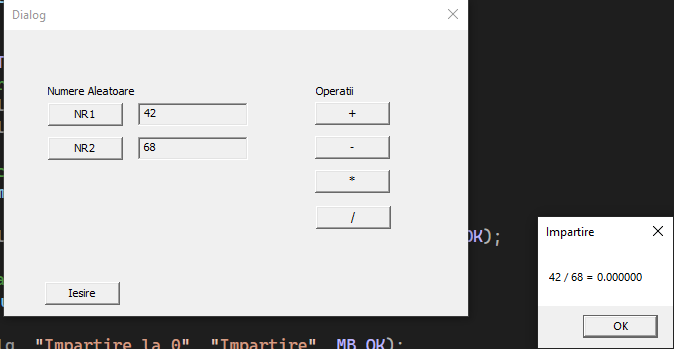
}











# Exercitiul 3

//#undef UNICODE

#include <windows.h>

#include <iostream>

#include <stdio.h>

#include "resource.h"

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE;

HWND hwndMain;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Dialog2";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwnd = CreateWindow(szAppName, // window class name

"Test dialog", // window caption

WS\_OVERLAPPEDWINDOW, // window style

CW\_USEDEFAULT, // initial x position

CW\_USEDEFAULT, // initial y position

CW\_USEDEFAULT, // initial x size

CW\_USEDEFAULT, // initial y size

NULL, // parent window handle

NULL, // window menu handle

hInstance, // program instance handle

NULL); // creation parameters

SetWindowPos(hwnd, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOACTIVATE);

ShowWindow(hwnd, SW\_HIDE);

UpdateWindow(hwnd);

hwndMain = hwnd;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

static HINSTANCE hInstance;

switch (iMsg)

{

case WM\_CREATE: //operatii ce se executa la crearea ferestrei

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) //dlgActive e false atunci

{// se creaza caseta de dialog

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, (DLGPROC)DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

//atasarea mesajului windows de închidere in coada de mesaje

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

/////////////////////////////////////////////////////////////

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

HWND hwnd;

char text[100] = "0";

int min, max;

min = 3;

max = 3;

int eur, dollar, ron, iCheck;

float eurCurrency = 4.34;

float dollarCurrency = 3.05;

float ronCurrency = 4.9;

//eurCurrency = roundf(eurCurrency \* 100) / 100.0;

//dollarCurrency = roundf(dollarCurrency \* 100) / 100.0;

//ronCurrency = roundf(ronCurrency \* 100) / 100.0;

static char userInput[100] = "\0";

static char buffer\_euro[100] = "\0";

static char buffer\_dollar[100] = "\0";

static char buffer\_ron[100] = "\0";

switch (iMsg) {

case WM\_INITDIALOG:

//Functia GetDlgItem obtine handler-ul unui control folosindu-se

// identificatorul (IDC\_CHECK3) acelui control

//validare control

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_RON);

SendMessage(hwnd, BM\_SETCHECK, (WPARAM)BST\_CHECKED, 0);

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_EUR);

SendMessage(hwnd, BM\_SETCHECK, (WPARAM)BST\_CHECKED, 0);

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_DOLLAR);

SendMessage(hwnd, BM\_SETCHECK, (WPARAM)BST\_CHECKED, 0);

// afisare curs valutar initial pentru euro

snprintf(buffer\_euro, sizeof buffer\_euro, "%f", eurCurrency);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_CURRENCY, buffer\_euro);

// afisare curs valutar initial pentru dolari

snprintf(buffer\_dollar, sizeof buffer\_dollar, "%f", dollarCurrency);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_CURRENCY, buffer\_dollar);

// Setare pe 0 a casetelor de editare

SetDlgItemText(hDlg, IDC\_EDIT\_RON\_RES, text);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_RES, text);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_RES, text);

SetDlgItemText(hDlg, IDC\_EDIT\_USERINPUT, text);

return TRUE;

case WM\_CLOSE:

if (MessageBox(hDlg, "Close the window?", "Close",

MB\_ICONQUESTION | MB\_YESNO) == IDYES)

{

DestroyWindow(hDlg);

}

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDC\_BUTTON\_CHANGE\_CURRENCY:

// schimbarea cursului valutar cu generare aleatorie: EURO

srand(time(NULL));

do

{

eurCurrency = (float)(rand() % min + max);

eurCurrency += ((float)rand() / RAND\_MAX);

// schimbarea cursului valutar si generarea unei valori DOLLAR mai mare decat cea EURO

do

{

dollarCurrency = (float)(rand() % min + max);

dollarCurrency += ((float)rand() / RAND\_MAX);

} while (dollarCurrency < eurCurrency);

} while (eurCurrency > 5 && dollarCurrency > 5);

snprintf(buffer\_euro, sizeof buffer\_euro, "%f", eurCurrency);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_CURRENCY, buffer\_euro);

snprintf(buffer\_dollar, sizeof buffer\_dollar, "%f", dollarCurrency);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_CURRENCY, buffer\_dollar);

break;

case IDC\_BUTTON\_CONVERT:

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_RON);

iCheck = (int)SendMessage(hwnd, BM\_GETCHECK, 0, 0);

// daca este selectat radio button pentru RON

if (iCheck)

{

// RON

GetDlgItemText(hDlg, IDC\_EDIT\_USERINPUT, userInput, 100);

SetDlgItemText(hDlg, IDC\_EDIT\_RON\_RES, userInput);

// EURO

GetDlgItemText(hDlg, IDC\_EDIT\_EUR\_CURRENCY, buffer\_euro, 100);

eurCurrency = atof(buffer\_euro);

eur = atof(userInput) / eurCurrency;

sprintf\_s(buffer\_euro, "%ld", eur);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_RES, buffer\_euro);

// DOLLAR

GetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_CURRENCY, buffer\_dollar, 100);

dollarCurrency = atof(buffer\_dollar);

dollar = atof(userInput) / dollarCurrency;

sprintf\_s(buffer\_dollar, "%ld", dollar);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_RES, buffer\_dollar);

}

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_EUR);

iCheck = (int)SendMessage(hwnd, BM\_GETCHECK, 0, 0);

// daca este selectat radio button pentru EURO

if (iCheck)

{

// EURO

GetDlgItemText(hDlg, IDC\_EDIT\_USERINPUT, userInput, 100);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_RES, userInput);

// RON

ron = atof(userInput) \* eurCurrency;

sprintf\_s(buffer\_ron, "%ld", ron);

SetDlgItemText(hDlg, IDC\_EDIT\_RON\_RES, buffer\_ron);

// DOLLAR

GetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_CURRENCY, buffer\_dollar, 100);

dollarCurrency = atof(buffer\_dollar);

dollar = atof(userInput) \* dollarCurrency;

sprintf\_s(buffer\_dollar, "%ld", dollar);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_RES, buffer\_dollar);

}

hwnd = GetDlgItem(hDlg, IDC\_RADIO\_DOLLAR);

iCheck = (int)SendMessage(hwnd, BM\_GETCHECK, 0, 0);

// daca este selectat radio button pentru DOLLAR

if (iCheck)

{

// DOLLAR

GetDlgItemText(hDlg, IDC\_EDIT\_USERINPUT, userInput, 100);

SetDlgItemText(hDlg, IDC\_EDIT\_DOLLAR\_RES, userInput);

// EURO

GetDlgItemText(hDlg, IDC\_EDIT\_EUR\_CURRENCY, buffer\_euro, 100);

eurCurrency = atof(buffer\_euro);

eur = atof(userInput) / eurCurrency;

sprintf\_s(buffer\_euro, "%ld", eur);

SetDlgItemText(hDlg, IDC\_EDIT\_EUR\_RES, buffer\_euro);

// RON

ron = atof(userInput) / ronCurrency;

sprintf\_s(buffer\_ron, "%ld", ron);

SetDlgItemText(hDlg, IDC\_EDIT\_RON\_RES, buffer\_ron);

}

break;

case IDC\_BUTTON\_EXIT:

MessageBox(hDlg, "Abandon!", "Ati selectat CANCEL", MB\_OK);

dlgActive = FALSE;

EndDialog(hDlg, 0);

return TRUE;

}

break;

}

return FALSE;

}

